TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

OPONENTNÍ POSUDEK ZÁVĚREČNÉ KVALIFIKAČNÍ PRÁCE

Autor závěrečné práce: Jaroslav Hrabal

Název práce: Objektově orientovaný framework pro provozní data podzemních zásobníků plynu

Oponent práce ing. Igor Kopetschke

Pracoviště oponenta Technická univerzita v Liberci

A. Kvalita abstraktu, klíčová slova odpovídají náplni práce	Velmi dobře (2)
B. Rozsah a zpracování rešerše	Velmi dobře mínus (2-)
C. Řešení práce po teoretické stránce	Dobře (3)
D. Vhodnost, přiměřenost použité metodiky	Velmi dobře mínus (2-)
E. Úroveň zpracování výsledků a diskuse	Nedostatečně (4)
F. Vlastní přínos k řešené problematice	Velmi dobře mínus (2-)
G. Formulace závěru práce	Dobře (3)
H. Splnění zadání (cílů) práce	Nesplněno
I. Skladba, správnost a úplnost citací literárních údajů	Velmi dobře mínus (2-)
J. Typografická a jazyková úroveň (vč. pravopisu)	Nedostatečně (4)
K. Formální náležitosti práce (struktura textu, řazení kapitol, přehlednost ilustrací)	Dobře (3)

Komentáře či připomínky:

Již při prvním čtení práce vykazuje znaky konceptu, nikoliv finálního produktu.
Po stránce syntaktické lze konstatovat, že autor evidentně svou práci nečetl (věty často začínají malým písmenem, místy jsou neúplné nebo nedávají smysl, vyskytují se slangové a hovorové výrazy).
Po stránce obsahové mám veliké výhrady od kapitoly 3 dále a jejich souhrn z důvodu rozsáhlosti uvedu v příloze. Výsledná implementace návrhu neodpovídá popisu uvedeném v práci - podrobnosti opět viz. příloha.
Přiložená aplikace je (snad) funkční za předpokladu přesného dodržení jednotlivých kroků, které nejsou formou dokumentace popsány. Aplikaci se podařilo po importu externích (k práci nepřiložených) knihoven zkompilovat a spustit, ale ani po hodině práce s ní jsem nedokázal získat kýžený výstup.

Celkově práce působí jako nedotažená a ve spěchu odbytá.

...pokračuje na straně 2

Celkové zhodnocení:

Jak jsem uvedl výše, podrobný komentář k jednotlivým výhradám přikládám v příloze. Konstatování v závěru, že byl vytvořen funkční famework, mi vzhledem k předloženému výsledku připadá značně nadnesený.
Dle mého názoru výsledek nesplňuje nároky kladené na bakalářskou práci a autor by ji měl přepracovat.

Otázky k obhajobě:

1. Četl jste svou práci před odevzdáním do tisku ?
2. Vysvětlete rozdíl mezi třídou Collections a rozhraním Collection z balíku java.util.*
3. Popište blíže strukturu souborů TBD a uveďte, jedná-li se o standardizovaný formát nebo ne
4. Proč je třída Virtual potomkem třídy LoadJSON a jak tato konstrukce zapadá do objektového modelu frameworku ?
5. Můžete jako součást prezentace odpovědí na tyto otázky ukázat UML diagram tříd Vašeho frameworku ?

Celková klasifikace:

Práce nesplňuje požadavky na udělení akademického titulu, a proto ji nedoporučuji k obhajobě

Navrhuji tuto práci klasifikovat stupněm Nedostatečně (4)

V Liberci

dne 2. června 2015

Podpisem současně potvrzuji, že nejsem v žádném osobním vztahu k autorovi práce

. . . . . .

podpis oponenta

Příloha oponentního posudku bakalářské práce

Autor práce: Jaroslav Hrabal

Název práce: Objektově orientovaný framework pro provozní data podzemních zásobníků plynu.

Oponent: ing. Igor Kopetschke

Podrobnější popis výhrad k práci

Jazyková, formální a syntaktická úroveň

Jak již bylo řečeno v posudku, autor psal práci evidentně ve spěchu a neprovedl následnou revizi textu. Věty často začínají malým písmenem, pojmenování některých proměnných a metod taktéž občas vzájemně neodpovídají skutečnosti. V textu se hojně vyskytují hovorové výrazy, například:

- strana 33 – "Výsledkem metod *all*, *group* a *wellNames* je **treemapa** s klíčem javovského času představující čas měření a **hodnoutou** v podobě listu double hodnot. "
- strana 34 – "... hodnoty ve tvaru **listů doubelu**"

Kompletní výčet pravopisných chyb a hovorových výrazů by rozsahem konkuroval celé práci.

Obsahová úroveň

Kapitola 3

Definovat OOP jako styl programování – zde bych asi našel lepší a přesnější formulaci. Nicméně tvrdit, že princip "*OOP spočívá v rozdělení programu na dílčí části a postupném řešení těchto menších částí programu*" je velice zavádějící, protože toto není unikátní vlastnost pouze OOP. Stejně tak není unikátem OOP znovupoužitelnost kódu.

Ze stejného důvodu mám výhrady k podkapitole 3.2 popisující princip dědičnosti.

Kapitola 3.6

Kapitola se věnuje tématu Java Collections Framework. Zcela chybí popis kolekcí nebo alespoň nástin hierarchie rozhraní a tříd. V první větě je sice konstatována existence **třídy Collections**, což ovšem není základní API jednotlivých rozhraní konkrétních typů kolekcí (tou je **rozhraní Collection**), ale nadstavbová třída poskytující algoritmy mj. pro práci s existujícími instancemi.

Následný popis jednotlivých rozhraní opět obsahuje chyby jak faktické, tak syntaktické – " Třída TreeMap implementuje rozhraní **map**."

Kapitola 5

Je zmiňován souborový formát TBD, ale nikde není popsáno, jedná-li se o uživatelem navržený formát, nebo formát standardizovaný, každopádně chybí jeho popis. Ten lze dovodit pouze z přiložených obrázků.

Kapitola 6

Tato kapitola a související podkapitoly popisují samotný objektový návrh frameworku. Struktura objektů popsaná v podkapitole 6.1 nerespektuje požadovanou hierarchii : **Virtual -> Storage -> WellGroup -> Well -> Quantity**

Chybějící třída WellGroup by měla představovat skupinu sond na zásobníku a obsahovat seznam jednotlivých sond. Naproti tomu zcela chybí a členství ve skupině je realizováno pouze atributem přímo v třídě Well. Tím se zbytečně komplikuje vyhledávání sond patřících do konkrétní skupiny.

Místo obrázku č. 7 by měl být uveden UML diagram tříd.

V kapitole 6.5 je popisován balíček main a třída Main, která dle popisu slouží k testování frameworku. Toto je velice zavádějící, protože standardně je pro tyto účely používána některá z metod testování SW, například JUnit testy, které poté logicky stojí mimo samotnou aplikaci.

Kapitola 8.2.2 nese název **Ukládání dat z datového souboru**, přitom popisuje načítání dat z TBD souborů a jejich zpracování.

Implementace frameworku

Zde jsou mé výhrady asi největší. V kódu se vyskytuje množství nepoužitých importů. Balíčky jsou pouze jednoúrovňové, což pro znovupoužitelnost jako framework není vhodné.

Některé třídy (například Well) obsahuje zcela nesmyslně atribut **cs** typu Scanner, navíc nepoužitý. Přitom tato třída má být dle návrhu třídou obalovou (bean) zapouzdřující pouze atributy sondy s přístupem pomocí metod get a set. Atributy navíc nesplňují princip zapouzdření, protože zcela chybí modifikátor private.

Třída Virtual měla reprezentovat dle zadání virtuální zásobník, mj jako seznam zásobníků reálných. V popisu v práci zcela schází. Navíc je potomkem třídy LoadJSON, jejímž účelem je pouze načtení zásobníku a souvisejících dat ze souboru typu json. Toto dědění je v kontextu účelu třídy Virtual z pohledu objektového modelu zcela chybné.

Mimochodem, v práci zcela schází zmínka o práci s datovým typem json.

Třída Main (výhrady viz výše) obsahuje kód s napevno deklarovanou absolutní cestou k souborům, kterou lze obejít pouze úpravou zdrojového kódu a nikoli např. parametrem při spuštění.

Přiložené datové soubory neobsahují jakýkoli příklad práce s json.

Zdrojové kódy jsou poskytnuty bez nutných knihoven třetích stran, které nejsou standardní součástí Javy (např. org.json.*), tudíž je bylo nutno před kompilací stáhnout. Kompilace proběhla v pořádku. Z důvodů výše uvedených nelze spustit testovací třídu Main bez nutných programových úprav.

Grafické rozhraní obsahuje nešťastně 4 spustitelné třídy, přičemž pouze jedna z nich umožní práci s aplikací. Bohužel, práce s aplikací je nepřehledná a ani po hodině testování se mi nepodařilo projít všemi úkony bez výskytu chyby.

Na rozdíl od tvrzení autora v práci, není ošetřena spousta stavů, které vyvolají výjimku – stačí zvolit soubor se špatnou strukturou a aplikace vypíše na konzoli chybové hlášení aniž je uživatel o tomto informován. Jakákoli nekonzistence v datech, zvoleném postupu, nadefinovaných kritériích atd. opět skončí výpisem chyby – ovšem nikoli výjimkou zachycenou programátorem, ale interní výjimkou Javy, kde uživatel nemá šanci zjistit příčinu výjimky v kontextu činnosti.

V Liberci dne 2.6.2015

Ing. Igor Kopetschke